Playing with Conway's Problem

Emmanuel Jeandel, Nicolas Ollinger

Playing with Conway's Problem

Emmanuel Jeandel [a,b], Nicolas Ollinger [b,*]

[a] *LIP, École Normale Supérieure de Lyon, CNRS*
46 allée d'Italie, 69007 Lyon, France

[b] *LIF, Aix-Marseille Université, CNRS,*
39 rue Joliot-Curie, 13013 Marseille, France

Abstract

The centralizer of a language is the maximal language commuting with it. The question, raised by Conway in 1971, whether the centralizer of a rational language is always rational, recently received a lot of attention. In Kunc 2005, a strong negative answer to this problem was given by showing that even complete co-recursively enumerable centralizers exist for finite languages. Using a combinatorial game approach, we give here an incremental construction of rational languages embedding any recursive computation in their centralizers.

In 1999, Choffrut *et al.* [1] renewed an old problem raised by Conway [2] in 1971: given a rational language, does its centralizer — the maximal language commuting with it — have to be rational? The property is known to hold for some particular families of languages. In the case of codes, Ratoandramanana [3] showed in 1989 that it holds for prefix codes, raising a restriction of Conway's problem to codes which recently recieved a positive answer by Karhumäki *et al.* [4]. In the general case, until recently, the best known result, by Karhumäki and Petre [5], was that the centralizer of a recursive language has to be co-recursively enumerable. This property may also be considered as a particular case of results of Okhotin [6] concerning the computational power of systems of equations on languages. For a complete survey on Conway's problem, the reader may refer to [7–10]. In 2004, the community was thrilled by an announcement by Kunc [11] that a centralizer can actually be non-recursive. This announcement was followed by a conference communication [12] in 2005 showing that finite languages exist whose centralizers are complete for co-recursively enumerable languages [1]. It includes a sketch of the proof for the special case of rational languages. While simpler than the proof

* Corresponding author.
Email address: Nicolas.Ollinger@lif.univ-mrs.fr (Nicolas Ollinger).

[1] Since the writing of the present paper, a journal version appeared in [13]

for finite languages, this proof is still rather involved — mostly due to a direct construction of the language encoding a given Minsky machine.

In this paper we propose another proof of the existence of rational languages with non-recursive centralizers. The key arguments of the proof come from a careful study of the first example in Kunc [12] leading to the core constructions of our proof: *checking* and *flooding*. Our approach significantly differs for two reasons. First, a combinatorial game point of view is taken through the whole proof. Games are convenient tools to embed a dynamical process like a computation into a static object like a fix-point. Using this point of view, a computation can be transformed incrementally into a centralizer by transforming winning strategies from one game to another more specialized game. Secondly, the construction of the language embedding a particular computation is incremental — explicitly explaining how to compile any program into a language so that its centralizer corresponds to the computation. Whereas the final proof is by no way shorter than Kunc original proof, cutting the construction into locally independent propositions improves its readability. Our proof also uses Post tag systems instead of Minsky machines as Post tag systems are in a way closer to centralizers.

In this paper, the letters Σ and Γ denote *finite alphabets*. The set of finite words over an alphabet Σ is denoted by $\Sigma^\star$, the *empty word* by ε, the *catenation* of two words x and y by xy and the length of $x \in \Sigma^\star$ by $|x|$. A word x is a *prefix* (resp. *suffix*) of a word y, if there exists a word $z \in \Sigma^\star$ such that $xz = y$ (resp. $y = zx$); this word z is unique and is denoted as $x^{-1}y$ (resp. yx^{-1}). A word x is a *subword* of a word y if there exist two words $z, z' \in \Sigma^\star$ such that $zxz' = y$. A *language* over Σ is a subset of $\Sigma^\star$. The *product* XY of two languages X and Y is the language $\{xy : x \in X, y \in Y\}$. The language of prefixes (resp. suffixes) of a language X, denoted as $\mathrm{Pref}(X)$ (resp. $\mathrm{Suff}(X)$), is the set of all prefixes (resp. suffixes) of words in X. The language of subwords of a language X, denoted as $\mathrm{Sub}(X)$, is the set of all subwords of words in X. The language $X^{-1}Y$ is the language $\{z : \exists x \in X, \exists y \in Y, y = xz\}$. The language YX^{-1} is the language $\{z : \exists x \in X, \exists y \in Y, y = zx\}$.

Two languages X and Y *commute* if the equation $XY = YX$ is satisfied. The set of languages that commute with a given language X is closed under infinite union. Thus it admits a unique maximal element for inclusion called the *centralizer* of X, denoted by $\mathcal{C}(X)$. The centralizer of X always contains $X^\star$. Moreover, if X contains the empty word then its centralizer is equal to $\Sigma^\star$. Otherwise, it is contained in $\mathrm{Pref}(X^\star) \cap \mathrm{Suff}(X^\star)$.

Conway's Problem Is $\mathcal{C}(X)$ rational if X is rational?

The paper is constructed as follows. In section 1, a particular family of games called cutenation games are introduced. These games can be viewed as an

extension of tag systems with states, languages constraints on states and the ability to cut and catenate on both sides of the word. In section 2, tag systems are encoded into games verifying some regularity properties. In section 3, these games are recursively transformed into games with only two states. In section 4, language constraints on both states are removed. In section 5, every parts are glued together to obtain the main result.

1 Cutenation games

In this section cutenation [2] games are introduced and their relations with centralizers are explained before sketching the proof of existence of rational languages with non-recursive centralizers.

1.1 Definition

A *cutenation game* is a tuple $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ where $\mathfrak{A}$ and $\mathfrak{B}$ are finite, both $(\mathfrak{A}, \mathfrak{B}, L)$ and $(\mathfrak{A}, \mathfrak{B}, R)$ are bipartite graphs whose edges are tagged with words on Σ (*i.e.* $L, R \subseteq \mathfrak{A} \times \mathfrak{B} \times \Sigma^\star$) and the mappings $V_A : \mathfrak{A} \to \mathrm{Rat}(\Sigma^\star)$ and $V_B : \mathfrak{B} \to \mathrm{Rat}(\Sigma^\star)$ constraint the positions. Given such a game, an *A-configuration* $(\mathfrak{a}, x) \in \mathfrak{A} \times \Sigma^\star$ verifies $x \in V_A(\mathfrak{a})$. A pair $(\mathfrak{a}, x) \in \mathfrak{A} \times \Sigma^\star$ might not be a valid position of the game. Symmetrically a *B-configuration* $(\mathfrak{b}, y) \in \mathfrak{B} \times \Sigma^\star$ verifies $y \in V_B(\mathfrak{b})$.

Remark. In this paper we will only consider connected cutenation games, that is cutenation games $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ for which the bipartite graph $(\mathfrak{A}, \mathfrak{B}, L \cup R)$ is connected.

Notation. We will depict L and R by a graph where $\mathfrak{A}$-vertices are represented by black points, $\mathfrak{B}$-vertices are represented by white points, L-edges are represented by plain edges and R-edges are represented by dashed edges (for clarity ε tags will be omitted).

Example. A sample cutenation game, omitting V_A and V_B, is depicted on Fig. 1 where $\mathfrak{A} = \{\alpha, \beta, \gamma, \delta\}$, $\mathfrak{B} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, $L = \{(\alpha, \mathrm{b}, \varepsilon), (\alpha, \mathrm{c}, \varepsilon), (\beta, \mathrm{a}, ab)\}$ and $R = \{(\alpha, \mathrm{a}, \varepsilon), (\beta, \mathrm{c}, \varepsilon), (\gamma, \mathrm{c}, \varepsilon), (\delta, \mathrm{b}, baa), (\delta, \mathrm{c}, \varepsilon)\}$.

A cutenation game is played as an iterated two-player combinatorial game where the set of A-configurations is the set of positions of the player A and

[2] *cutenation* is a free contraction of both words *cut* and *catenation*.

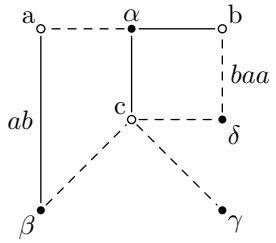

Fig. 1. graphical representation of a simple cutenation game

the set of B-configurations is the set of positions of the player B. A move of the player A, from an A-configuration $(\mathfrak{a}, x)$ to a B-configuration $(\mathfrak{b}, y)$, is a catenation:

- either an l-move $(\mathfrak{a}, x) \vdash_{A,l} (\mathfrak{b}, zx)$ such that $y = zx$ and $(\mathfrak{a}, \mathfrak{b}, z) \in L$;
- or an r-move $(\mathfrak{a}, x) \vdash_{A,r} (\mathfrak{b}, xz)$ such that $y = xz$ and $(\mathfrak{a}, \mathfrak{b}, z) \in R$.

Symmetrically, a move of the player B, from a B-configuration $(\mathfrak{b}, y)$ to an A-configuration $(\mathfrak{a}, x)$, is a cut:

- either an l-move $(\mathfrak{b}, zx) \vdash_{B,l} (\mathfrak{a}, x)$ such that $y = zx$ and $(\mathfrak{a}, \mathfrak{b}, z) \in L$;
- or an r-move $(\mathfrak{b}, xz) \vdash_{B,r} (\mathfrak{a}, x)$ such that $y = xz$ and $(\mathfrak{a}, \mathfrak{b}, z) \in R$.

A round of the game starts from an A-configuration $(\mathfrak{a}, x)$ and consists first of a move of the player A from $(\mathfrak{a}, x)$ to a B-configuration $(\mathfrak{b}, y)$, then of a move of the player B from $(\mathfrak{b}, y)$ to an A-configuration $(\mathfrak{a}', x')$. Furthermore, if A plays an l-move then B must play an r-move and symmetrically if A plays an r-move then B must play an l-move. If a player cannot move then the player loses. The next round will start from $(\mathfrak{a}', x')$. If the game lasts forever then the player B wins.

Example. For the cutenation game of Fig. 1, this is a valid sequence of consecutive rounds, assuming $V_A(\mathfrak{a})$ and $V_B(\mathfrak{b})$ always equal to $\Sigma^\star$:

$$\begin{array}{lllll} 1.\ (\beta, aa) & \vdash_{A,l} & (\mathrm{a}, abaa) & \vdash_{B,r} & (\alpha, abaa)\ ; \\ 2.\ (\alpha, abaa) & \vdash_{A,l} & (\mathrm{b}, abaa) & \vdash_{B,r} & (\delta, a)\ ; \\ 3.\ (\delta, a) & \vdash_{A,r} & (\mathrm{c}, a) & \vdash_{B,l} & (\alpha, a)\ . \end{array}$$

Notice that such a game can be played without memory. Thus a *strategy* for the player A in this game is simply a mapping from A-configurations to valid moves from the given configuration. The strategy is winning for a given configuration if, when the player A plays according to the strategy, whatever moves the player B decide to play, the player A wins the game. Strategies and winning strategies for the player B are defined symmetrically. A configuration

is called a winning position for a given player if there exists a winning strategy from this configuration for this player. The following classical result holds for these games, we give here a sketch of a proof.

Lemma 1 *Starting from an A-configuration $(\mathfrak{a}, x)$ either the player A has a winning strategy or the player B has a winning strategy.*

PROOF. Let $(\mathfrak{a}, x)$ be an A-configuration for which neither the player A nor the player B has a winning strategy. If every move from the player A starting from $(\mathfrak{a}, x)$ would lead to a B-configuration from which the player B could move to an A-configuration on which the player B has a winning strategy then the position $(\mathfrak{a}, x)$ would be winning for the player B. Thus, the player A has a valid move from $(\mathfrak{a}, x)$ to a B-configuration $(\mathfrak{b}, y)$ from which the player B can move either to A-configurations on which the player A has a winning strategy or to A-configurations on which neither the player A nor the player B has a winning strategy. On such configurations the best moves from both the player A and the player B would lead to an infinite run. By the rules, the player B would win which implies that the player B has a winning strategy starting from $(\mathfrak{a}, x)$. □

1.2 Languages and centralizers

Given a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ and an element $\mathfrak{a}$ of $\mathfrak{A}$, the language $\mathfrak{L}(\mathfrak{a})$ is the set of words $x \in \Sigma^\star$ such that the configuration $(\mathfrak{a}, x)$ admits a winning strategy for the player B.

In the special case where L, R, V_A and V_B are recursive, given an element $\mathfrak{a}$ of $\mathfrak{A}$, the language $\mathfrak{L}(\mathfrak{a})$ is co-recursively enumerable. It follows from the fact that one can exhaustively search a winning strategy for the player A as a finite one exists — the player B only has finitely many valid moves starting from a B-configuration, as one word has finitely many subwords.

The centralizer $\mathcal{C}(X)$ of a given language X can be expressed as the language $\mathfrak{L}$ associated with the unique element of $\mathfrak{A}$ of the cutenation game where both $\mathfrak{A}$ and $\mathfrak{B}$ are singletons, L and R are both equal to the language X, and both $V_A(\mathfrak{a}) = \Sigma^\star$ and $V_B(\mathfrak{b}) = \Sigma^\star$. In the following, we call such a game a commutation game. For the sake of readability, when manipulating cutenation game where $\mathfrak{A}$ and $\mathfrak{B}$ are singletons, we will manipulate L, R, V_A and V_B as subsets of $\Sigma^\star$ and denote the language associated with the game as $\mathfrak{L}$. For the same reasons A-configurations and B-configurations will be considered as elements of $\Sigma^\star$.

In order to prove the main result of this paper, we will proceed through the following steps. First, we restrict ourselves to a specific subset of special cutenation games. Then, we show how to recursively encode co-recursively enumerable languages into the language of such a game. After that we proceed to the core of the proof and explain how to transform such special cutenation game into a commutation game. During this transformation, the language associated with any element of $\mathfrak{A}$ is recursively encoded into the language associated with the commutation game of a rational language.

2 Encoding Post Tag Systems

In order to encode every co-recursively enumerable language into the language associated with a commutation game, the family of cutenation games is first restricted to games with special properties that will allow further reductions; then Post tag systems are encoded into games verifying these particular properties.

2.1 Restraining Cutenation Games

The following special kinds of cutenation games will be used in the proof. The main reason to enforce these properties is to enable the later encoding of both $\mathfrak{A}$ and $\mathfrak{B}$ into L, R, V_A and V_B.

Unfairness. A cutenation game is *unfair* if the player A has no constraint. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ over the alphabet Σ is *unfair* if for all $\mathfrak{b} \in \mathfrak{B}$, $V_B(\mathfrak{b}) = \Sigma^\star$.

Rootedness. A cutenation game is *rooted* if the player A can catenate non-empty words on the left (respectively on the right) from at most one position called the left root (respectively the right root). More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *rooted* if there exists a left root $\mathfrak{a}_L \in \mathfrak{A}$ such that for all $(\mathfrak{a}, \mathfrak{b}, x) \in L$ if $x \neq \varepsilon$ then $\mathfrak{a} = \mathfrak{a}_L$ and there exists a right root $\mathfrak{a}_R \in \mathfrak{A}$ such that for all $(\mathfrak{a}, \mathfrak{b}, x) \in R$ if $x \neq \varepsilon$ then $\mathfrak{a} = \mathfrak{a}_R$.

Oscillation. A cutenation game is *oscillating* if the player A is enforced to oscillate at each round between l-moves and r-moves. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *oscillating* if the set $\mathfrak{A}$ can be split into two disjoint sets $\mathfrak{A}_L$ and $\mathfrak{A}_R$ such that for all $(\mathfrak{a}, \mathfrak{b}, x) \in L$ necessarily $\mathfrak{a} \in \mathfrak{A}_L$ and for all $(\mathfrak{a}, \mathfrak{b}, x) \in R$ necessarily $\mathfrak{a} \in \mathfrak{A}_R$.

SEPARATION. A cutenation game is *separated* if positions can be viewed as pairs of left and right positions, a left position being only affected by l-moves and a right position only by r-moves. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *separated* if there exist two sets S_L and S_R such that $\mathfrak{A} \cup \mathfrak{B} \subseteq S_L \times S_R$ and both L and R satisfy the following requirements. For every move $((s,t),(s',t'),x) \in L$ only the left part is modified, so $t = t'$. Moreover, for every $t'' \in S_R$ such that $(s,t'') \in \mathfrak{A}$ necessarily $(s',t'') \in \mathfrak{B}$ and the move $((s,t''),(s',t''),x)$ must be in L. Symmetrically, for every move $((s,t),(s',t'),x) \in R$ only the right part is modified, so $s = s'$. Moreover, for every $s'' \in S_L$ such that $(s'',t) \in \mathfrak{A}$ necessarily $(s'',t') \in \mathfrak{B}$ and the move $((s'',t),(s'',t'),x)$ must be in R.

ORIENTATION. A cutenation game is *oriented* if it is both separated and oscillating and if its left and right positions can be ordered into minimal and maximal positions, a move changing the corresponding position from minimal to maximal. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *oriented* if it is both separated and oscillating and if the set S_L, respectively S_R, can be split into two disjoint sets S_L^- and S_L^+, respectively S_R^- and S_R^+, such that $\mathfrak{A}_L \subseteq S_L^- \times S_R^+$, $\mathfrak{A}_R \subseteq S_L^+ \times S_R^-$, and $\mathfrak{B} \subseteq S_L^+ \times S_R^+$.

Lemma 2 *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be an oscillating cutenation game. Let ν_L, symmetrically ν_R, be the function mapping an element of $\mathfrak{A} \cup \mathfrak{B}$ to its connected component in the bipartite graph $(\mathfrak{A}, \mathfrak{B}, L)$, symmetrically $(\mathfrak{A}, \mathfrak{B}, R)$. If the mapping $\nu : \mathfrak{a} \mapsto (\nu_R(\mathfrak{a}), \nu_L(\mathfrak{a}))$ is injective then the given oscillating cutenation game can be considered, up to the isomorphism ν, as a separated oscillating cutenation game where $S_L = \nu_R(\mathfrak{A} \cup \mathfrak{B})$ and $S_R = \nu_L(\mathfrak{A} \cup \mathfrak{B})$.*

PROOF. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be an oscillating cutenation game satisfying the hypothesis. Let $(\mathfrak{a}, \mathfrak{b}, x)$ be in L and let both $(s,t) = \nu(\mathfrak{a})$ and $(s',t') = \nu(\mathfrak{b})$. By definition of ν_L, as $\mathfrak{a}$ and $\mathfrak{b}$ are connected by L then $\nu_L(\mathfrak{a}) = \nu_L(\mathfrak{b})$ thus $t = t'$. Moreover, as the game is oscillating $\mathfrak{a} \in \mathfrak{A}_L$. Let $t'' \in S_R$ be such that $(s,t'') = \nu(\mathfrak{a}')$ for some $\mathfrak{a}' \in \mathfrak{A}$. By definition of ν_R this means that $\mathfrak{a}$ and $\mathfrak{a}'$ are connected by R. As $\mathfrak{a} \in \mathfrak{A}_L$ it implies that $\mathfrak{a} = \mathfrak{a}'$. A symmetrical reasoning applies to R. Therefore, the game is, up to the isomorphism ν, separated. □

Lemma 3 *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be a separated oscillating cutenation game obtained by Lemma 2. Such a game is oriented.*

PROOF. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be a separated oscillating cutenation game obtained by Lemma 2. Let S_L^+ be defined as $\{s : \exists t \in S_R, (s,t) \in \mathfrak{B}\}$ and $S_L^- = S_L \setminus S_L^+$. Symmetrically, let S_R^+ be defined as $\{t : \exists s \in S_L, (s,t) \in \mathfrak{B}\}$ and $S_R^- = S_R \setminus S_R^+$. By construction, the inclusion $\mathfrak{B} \subseteq S_L^+ \times S_R^+$ holds. Let (s,t)

be in $\mathfrak{A}_L$. As the cutenation game is connected there is at least one move in L involving (s,t) thus $t \in S_R^+$. Assume that $s \in S_L^+$. This means that there exists some $t' \in S_R^+$ such that $(s,t') \in \mathfrak{B}$. By definition of ν_R necessarily (s,t) and (s,t') are connected by R. As the game is oscillating it implies that $t = t'$, but $\mathfrak{A}$ and $\mathfrak{B}$ are disjoint. This is a contradiction, therefore s must be in S_L^-. Symmetrically, the same holds for $\mathfrak{A}_R$. □

2.2 Post Tag Systems

A *Post tag system* $\mathcal{P}$ is a triple (Σ, k, φ) where Σ is a finite alphabet, k is a positive integer and φ is a mapping from Σ^k to $\Sigma^\star$. A configuration of the system is a word from $\Sigma^\star$. For all y in $\Sigma^\star$ and i in Σ^k, the configuration iy evolves into the configuration $y\varphi(i)$. The computation stops when no further evolution is possible, *i.e.* when the length of the word is less than k. The language $L_{\mathcal{P}}$ associated with the Post tag system is the set of words for which the evolution eventually stops. Post tag systems are universal in the sense that one can recursively encode any recursively enumerable language into their languages. For more details about tag systems and their computational power, the reader might consult Minsky [14].

Proposition 4 *Let $\mathcal{P}$ be a Post tag system over the alphabet Σ. There exists an unfair rooted oriented cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ over the same alphabet such that, for some distinguished element $\mathfrak{a} \in \mathfrak{A}$, the languages $\mathfrak{L}(\mathfrak{a})$ and $\Sigma^\star \setminus L_{\mathcal{P}}$ are equal.*

PROOF. Let $\mathcal{P}$ be a Post tag system (Σ, k, φ). The tag system will be encoded as a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ where

$$\mathfrak{A} = \{\alpha, \eta\} \cup \bigcup_{i \in \Sigma^k} \{\beta_i, \gamma_i, \delta_i, \zeta_i\},$$
$$\mathfrak{B} = \{\mathrm{a}\} \cup \bigcup_{i \in \Sigma^k} \{\mathrm{b}_i, \mathrm{c}_i, \mathrm{d}_i\}$$

and the relations L and R are depicted on Fig. 2.

The constraints V_A are defined as follows: $V_A(\alpha) = \Sigma^\star$, $V_A(\eta) = \Sigma^\star$, and for all i in Σ^k:

$$V_A(\beta_i) = \left(\Sigma^k \setminus \{i\}\right) \Sigma^\star \varphi(i),$$
$$V_A(\gamma_i) = i\Sigma^\star \varphi(i),$$
$$V_A(\delta_i) = i\Sigma^\star \varphi(i),$$
$$V_A(\zeta_i) = \Sigma^\star \setminus i\Sigma^\star \varphi(i).$$

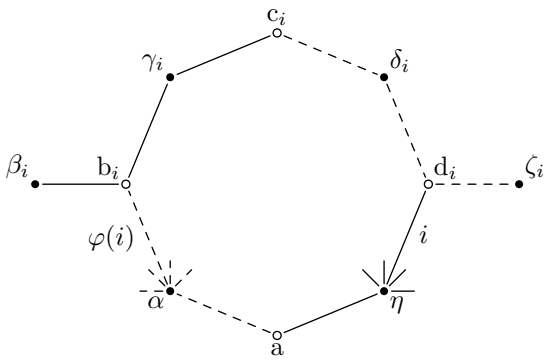

Fig. 2. the rooted oriented cutenation game of a Post system

This game is unfair and rooted, the roots being α and η. Moreover, it is oscillating and fulfills the requirements of Lemma 2 thus by Lemma 3 it is oriented. It remains to prove that $\mathfrak{L}(\alpha)$ equals $\Sigma^\star \setminus L_{\mathcal{P}}$.

Let x be a word in $L_{\mathcal{P}}$. A winning strategy for the player A starting from the A-configuration (α, x) is to follow the computation steps of the Post tag system. If a transition of the tag system exists starting from x then x can be written as iy with $i \in \Sigma^k$. Going through the states $\mathrm{b_i}$, $\mathrm{c_i}$, $\mathrm{d_i}$ and a, the player A will force the player B to go to the A-configuation $(\alpha, y\varphi(i))$. If no transition of the tag system exists starting from x, this means that $|x|$ is less than k, the player A moves to $\mathrm{b_i}$ for any $i \in \Sigma^k$. The player B has no valid move. The player A wins. Therefore, the player A has a winning strategy starting from (α, x) with $x \in L_{\mathcal{P}}$.

Let x be a word in $\Sigma^\star \setminus L_{\mathcal{P}}$. A winning strategy for the player B starting from the A-configuration (α, x) works as follows. In this game the player B has no choice so his strategy is to play when he can. The only possibility for the player B to have no valid move is to play from some position $\mathrm{b_i}$ obtained from the position α with a word of length less than k. Observe that the only possible sequences of moves going from a configuration (α, y) to a configuration (α, z) imply that in the tag system there is a valid sequence of forward and backward transitions from y to z. Thus, as in the tag system x has an infinite sequence of valid forward transitions, the position (α, x) is winning for the player B. Therefore, the player B has a winning strategy starting from (α, x) with $x \in \Sigma^\star \setminus L_{\mathcal{P}}$. $\square$

3 Removing states

In order to transform unfair rooted oriented cutenation games into commutation games, the first step is to transform the state sets $\mathfrak{A}$ and $\mathfrak{B}$ into singletons and to ensure that $L = R$. This is done by choosing a proper encoding of every configuration $((s, t), x)$ into a proper word $\langle s, x, t\rangle$.

3.1 Encoding states

Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be an unfair rooted oriented cutenation game. We encode each configuration $((s,t),x)$ into a proper word $\langle s,x,t\rangle$ using the following encoding.

Let m be the size of S_L^- and Γ_L^- be an alphabet of $m-1$ ordered new letters $\{\alpha_1,\ldots,\alpha_{m-1}\}$. Let ρ map S_L^- into $\{0,1,\ldots,m-1\}$ so that the left coordinate of the left root is mapped into 0. Let φ_L^- map $s \in S_L^-$ into the word $\alpha_{\rho(s)}\cdots\alpha_2\alpha_1$ of length $\rho(s)$. The encoding $\varphi_L(s)$ of a state $s \in S_L$ is equal to $\varphi_L^-(s)$ when $s \in S_L^-$. Let n be the size of S_L^+ and Γ_L^+ be an alphabet of n ordered new letters $\{\beta_1,\ldots,\beta_n\}$. Let σ map S_L^+ into $\{1,\ldots,n\}$. Let φ_L^+ map $s \in S_L^+$ into the word $\beta_{\sigma(s)}\alpha_{m-1}\cdots\alpha_1$ of length m. The encoding $\varphi_L(s)$ of a state $s \in S_L$ is equal to $\varphi_L^+(s)$ when $s \in S_L^+$. For each pair of states $(s,s') \in S_L^- \times S_L^+$ define $\phi_L(s,s')$ as $\varphi_L^+(s')\varphi_L^-(s)^{-1}$, which is $\beta_{\sigma(s')}\alpha_{m-1}\cdots\alpha_{\rho(s)+1}$. Notice that $\phi_L(s,s') \in \Gamma_L^+\left(\Gamma_L^-\right)^\star$.

Symmetrically, let m' be the size of S_R^- and Γ_R^- be an alphabet of $m'-1$ ordered new letters $\{\gamma_1,\ldots,\gamma_{m'-1}\}$. Let ρ' map S_R^- into $\{0,1,\ldots,m'-1\}$ so that the right coordinate of the right root is mapped into 0. Let φ_R^- map $t \in S_R^-$ into the word $\gamma_1\gamma_2\cdots\gamma_{\rho'(t)}$ of length $\rho'(t)$. The encoding $\varphi_R(t)$ of a state $t \in S_R$ is equal to $\varphi_R^-(t)$ when $t \in S_R^-$. Let n' be the size of S_R^+ and Γ_R^+ be an alphabet of n' ordered new letters $\{\delta_1,\ldots,\delta_{n'}\}$. Let σ' map S_R^+ into $\{1,\ldots,n'\}$. Let φ_R^+ map $t \in S_R^+$ into the word $\gamma_1\cdots\gamma_{m'-1}\delta_{\sigma'(t)}$ of length m'. The encoding $\varphi_R(t)$ of a state $t \in S_R$ is equal to $\varphi_R^+(t)$ when $t \in S_R^+$. For each pair of states $(t,t') \in S_R^- \times S_R^+$ define $\phi_R(t,t')$ as $\varphi_R^-(t)^{-1}\varphi_R^+(t')$, which is $\gamma_{\rho'(t)+1}\cdots\gamma_{m'-1}\delta_{\sigma'(t')}$. Notice that $\phi_R(t,t') \in \left(\Gamma_R^-\right)^\star\Gamma_R^+$.

Let τ_L and τ_R be the two morphisms from $\Sigma^\star$ to $(\Sigma\cup\{o\})^\star$, where o is a new letter, defined for each letter $a \in \Sigma$ by $\tau_L(a) = oa$ and $\tau_R(a) = ao$. For each word $x \in \Sigma^\star$ define $\tau(x)$ as $\tau_L(x)o$, which is equal to $o\tau_R(x)$. These morphisms will be used to encode configurations of the game with two goals in mind: first of all, ensure that no word is encoded as the empty word; secondly ensure that each encoded word has an odd length.

A configuration $((s,t),x) \in (S_L \times S_R)\times\Sigma^\star$ of the game will be encoded by the word $\varphi_L(s)\tau(x)\varphi_R(t)$ denoted as $\langle s,x,t\rangle$. The set L will be encoded using the mapping ψ_L defined by $\psi_L((s,t),(s',t),x) = \phi_L(s,s')\tau_L(x)$. Symmetrically, the set R will be encoded using the mapping ψ_R defined by $\psi_R((s,t),(s,t'),y) = \tau_R(y)\phi_R(t,t')$.

Remark. To summarize, $\mathfrak{A}_L$-configurations are encoded by words in the language $\mathrm{Suff}(\alpha_{m-1}\cdots\alpha_1)\,(o\Sigma)^\star o\gamma_1\cdots\gamma_{m'-1}\Gamma_R^+$, symmetrically $\mathfrak{A}_R$-configurations

are encoded by words in $\Gamma_L^+\alpha_{m-1}\cdots\alpha_1\,(o\Sigma)^\star o\,\mathrm{Pref}(\gamma_1\cdots\gamma_{m'-1})$, and finally, $\mathfrak{B}$-configurations are encoded by words in $\Gamma_L^+\alpha_{m-1}\cdots\alpha_1\,(o\Sigma)^\star o\gamma_1\cdots\gamma_{m'-1}\Gamma_R^+$.

Proposition 5 *Let $(\mathfrak{A},\mathfrak{B},L,R,V_A,\Sigma^\star)$ be an unfair rooted oriented cutenation game. Let V'_A be the set $\{\langle s,x,t\rangle : (s,t)\in\mathfrak{A}, x\in V_A((s,t))\}$ and V'_B the set $\mathrm{Sub}\left(\left\{\langle s,x,t\rangle : x\in\Sigma^\star,(s,t)\in S_L^+\times S_R^+\right\}\right)$. Let $((s,t),x)$ be an A-configuration of the game. There exists a valid l-move for the player A in the cutenation game $(\{\mathfrak{a}\},\{\mathfrak{b}\},\psi_L(L),\psi_R(R),V'_A,V'_B)$ from the configuration $\langle s,x,t\rangle$ to a configuration w if and only if $w=\langle s',y,t'\rangle$ for some s', y, t' and the l-move from $((s,t),x)$ to $((s',t'),y)$ is valid for the player A in the first game. The same holds for r-moves and B-configurations.*

PROOF. Let $(\mathfrak{A},\mathfrak{B},L,R,V_A,\Sigma^\star)$ be an unfair rooted oriented cutenation game. Let $((s,t),x)$ be a configuration of the game.

Let $((s',t'),y)$ be a configuration of the game such that a move from $((s,t),x)$ to $((s',t'),y)$ is valid. Let $w=\langle s',y,t'\rangle$. If the move is an l-move for the player A then $t=t'$ and $((s,t),(s',t),z)\in L$ where $y=zx$, and thus $\phi_L(s,s')\tau_L(z)\in\psi_L(L)$. To prove that this move is a valid l-move in the new game, it is sufficient to show that $\phi_L(s,s')\tau_L(z)\,\langle s,x,t\rangle=\langle s',y,t'\rangle$. If (s,t) is the left root then $\varphi_L(s)=\varepsilon$ and $\phi(s,s')=\varphi_L(s')$ thus $\phi_L(s,s')\tau_L(z)\,\langle s,x,t\rangle=\varphi_L(s')\tau(zx)\varphi_R(t)$. If (s,t) is not the left root then $z=\varepsilon$, and therefore $\phi_L(s,s')\tau_L(z)\,\langle s,x,t\rangle=\varphi_L(s')\tau(x)\varphi_R(t)$ as $\phi_L(s,s')\varphi_L(s)=\varphi_L(s')$. Therefore, if the move is a valid l-move for the player A in the original game then it is a valid l-move for the player A in the new game. The three other cases work on the same principle (do not forget to check with V_A in the case of a move for the player B).

Let w be a word such that there is a valid move for the player A in the new game from $\langle s,x,t\rangle$ to w where $(s,t)\in\mathfrak{A}$. If it is an l-move, there exist some s', s'' and z such that $\phi_L(s',s'')\tau_L(z)\in\psi_L(L)$ and $w=\phi_L(s',s'')\tau_L(z)\,\langle s,x,t\rangle$. As $w\in V'_B$ and both $s''\in S_L^+$ and $t\in S_R^+$ then $w=\langle s'',y,t\rangle$ for some $y\in\Sigma^\star$. This implies that $s=s'$ and $y=zx$. To prove that there is a valid l-move in the original game for the player A from the configuration $((s,t),x)$ to the configuration $((s'',t),zx)$ it is sufficient to show that $(s'',t)\in\mathfrak{B}$ and $((s,t),(s'',t),z)\in L$. As $\phi_L(s,s'')\tau_L(z)\in\psi_L(L)$ there exists some t' such that $((s,t'),(s'',t'),z)\in L$. As the original game is separated and both $(s,t)\in\mathfrak{A}$ and $(s,t')\in\mathfrak{A}$ then $(s'',t)\in\mathfrak{B}$ and $((s,t),(s'',t),z)\in L$. The case of an r-move for the player A works symmetrically.

Let w be a word such that there is a valid move for the player B in the new game from $\langle s,x,t\rangle$ to w where $(s,t)\in\mathfrak{B}$. If it is an l-move then there exists some s', s'' and z such that $\phi_L(s',s'')\tau_L(z)\in\psi_L(L)$ and $\phi_L(s',s'')\tau_L(z)w=\langle s,x,t\rangle$. As $w\in V'_A$ and both $s\in S_L^+$ and $t\in S_R^+$ then $s''=s$ and $w=\langle s',y,t\rangle$

where $(s',t) \in \mathfrak{A}$ and $y \in V_A((s',t))$ is such that $zy = x$. To prove that there is a valid l-move in the original game for the player B from the configuration $((s,t), zy)$ to the configuration $((s',t), y)$ it is sufficient to show that $(s',t) \in \mathfrak{A}_L$ and $((s',t),(s,t),z) \in L$. As $\phi_L(s',s)\tau_L(z) \in \psi_L(L)$ there exists some t' such that $((s',t'),(s,t'),z) \in L$. As the original game is separated and both $(s',t) \in \mathfrak{A}$ and $(s',t') \in \mathfrak{A}$ then $((s',t),(s,t),z) \in L$. The case of r-move for the player B works symmetrically. □

3.2 Enforcing symmetry

In a commutation game both sets of left moves L and right moves R are equal. If the sets V_A and V_B bring enough constraints to the game, both L and R can be replaced by $L \cup R$ to enforce this symmetry.

Proposition 6 *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, L, R, V_A, V_B)$ be a cutenation game. Let X be the language $L \cup R$. If the four sets $RV_A \cap V_B$, $V_AL \cap V_B$, $R^{-1}V_B \cap V_A$ and $V_BL^{-1} \cap V_A$ are empty then the valid moves, both for the player A and the player B, are the same in the given game and in the cutenation game $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V_A, V_B)$.*

PROOF. Every move in the original game is allowed in the new game. Conversely, let $x \vdash_{A,l} y$ be a valid l-move for the player A in the new game. There exists $z \in X$ such that $y = zx$ so $y \in XV_A \cap V_B$. As $RV_A \cap V_B$ is empty, then $z \in L$ and the move is also valid in the original game. The three remaining cases are similar using the three other empty sets (use $V_AL \cap V_B$ for $\vdash_{A,r}$, use $R^{-1}V_B \cap V_A$ for $\vdash_{B,l}$, and use $V_BL^{-1} \cap V_A$ for $\vdash_{B,r}$). □

The construction to enforce symmetry can be applied directly after encoding the states as the new encoding verifies the required hypothesis.

Lemma 7 *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ be some unfair rooted oriented cutenation game. Let V'_A and V'_B be defined as in Prop. 5. Let X be the set $\psi_L(L) \cup \psi_R(R)$. The valid moves for both the player A and the player B are the same in both games $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, \psi_L(L), \psi_R(R), V'_A, V'_B)$ and $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V'_A, V'_B)$.*

PROOF. By Prop. 6 it is sufficient to show that the four sets $\psi_R(R)V'_A \cap V'_B$, $V'_A\psi_L(L) \cap V'_B$, $\psi_R(R)^{-1}V'_B \cap V'_A$, and $V'_B\psi_L(L)^{-1} \cap V'_A$ are empty.

The language $\psi_R(R)V'_A$ does not intersect V'_B because every word of $\psi_R(R)V'_A$ contains an occurrence of a letter in Γ_R^+ before a letter o and this is never the case in V'_B. A symmetrical proof works for $V'_A\psi_L(L) \cap V'_B$.

The language $\psi_R(R)^{-1}V'_B$ does not intersect V'_A because $\psi_R(R)^{-1}V'_B$ only contains the empty word which is not in V'_A. The same holds for $V'_B\psi_L(L)^{-1}$. □

4 Removing constraints

In order to conclude the construction the constraints sets V_A and V_B must be removed. This part is the core of the proof. The construction proceeds in two steps : first we remove V_A through *checking*, then we remove V_B through *flooding*.

4.1 Checking

To remove V_A means to remove constraints on the positions at the end of a move from the player B. To ensure that the set of winning strategies of the player B does not grow, the idea is to allow the player A to challenge the player B if he plays outside of V_A by *checking* the validity of the move.

Proposition 8 *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V_A, V_B)$ be a cutenation game over the alphabet Σ with associated language $\mathfrak{L}$. Let c be a new letter not in Σ, let $X' = X \cup cV_A^\star \cup V_A^\star c$ and $V'_B = V_B \cup cV_B \cup V_B c$. Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X', X', (\Sigma \cup \{c\})^\star, V'_B)$ be the cutenation game over the alphabet $\Sigma \cup \{c\}$ with associated language $\mathfrak{L}'$. If the four sets $X^{-1}X^{-1}V_B$, $V_BX^{-1}X^{-1}$, $((X^{-1}(V_AX \cap V_B)) \setminus V_A) \cap V_A^\star$, and $(((XV_A \cap V_B)X^{-1}) \setminus V_A) \cap V_A^\star$ are empty and both inclusions $X^{-1}V_B \subseteq V_B$ and $V_BX^{-1} \subseteq V_B$ hold then $\mathfrak{L}$ is equal to $\mathfrak{L}' \cap \Sigma^\star$.*

PROOF. We prove that the player B has a winning strategy in the original game if and only if the player B has a winning strategy in the new game.

If the player B had a winning strategy in the original game starting from a given position then he keeps playing according to his original strategy as long as the player A keeps using moves that were valid in the original game. If the player A uses a new move from a position $x \in V_A$ then there are two possibilities:

- either he catenates a word of X, leading to a new position in $V'_B \setminus V_B$; this is impossible as all the new valid positions must contain the new letter c which does not appear in X;
- or he catenates a word of $X' \setminus X$ containing the new letter c, leading to a new valid position y which must be either in cV_B or in V_Bc; as $x \in V_A$ and as $X' \setminus X = cV_A^\star \cup V_A^\star c$, necessarily $y \in cV_A^\star \cup V_A^\star c$ and thus $y \in X'$.

A winning strategy for the player B starting from a position $y \in X'$ is simply to cut y completely thus leading to the empty word position. The empty word position is winning for the player B: when the player A catenates a word z, the player B just cuts z, coming back to the empty word. Therefore, the player B still has a winning strategy in the new game.

If the player A had a winning strategy in the original game starting from a given position then he keeps playing according to his original strategy as long as the player B keeps using moves that were valid in the original game. If the player B uses a new move from a position $x \in V_B$ then there are two possibilities:

- either he cuts a word of $X' \setminus X$ containing the new letter c; this is impossible as the new letter c does not appear in V_B;
- or he cuts a word of X, leading to a new valid position in $\Sigma^\star \setminus V_A$, more precisely in $((X^{-1}(V_A X \cap V_B)) \cup ((X V_A \cap V_B) X^{-1})) \setminus V_A$.

A winning strategy for the player A starting from a position y in the language $(X^{-1}(V_A X \cap V_B)) \setminus V_A$ is simply to catenate the word c on the right, leading to the valid position yc in $V_B c$. As $y \notin V_A^\star$ and $X^{-1}X^{-1}V_B = \emptyset$ the player B has no valid move starting from yc, thus the player A wins. By a symmetrical argument, the player A has a winning strategy starting from a position in $((X V_A \cap V_B) X^{-1}) \setminus V_A$. Therefore, the player A still has a winning strategy in the game. □

4.2 Flooding

To remove V_B means to remove constraints on the positions at the end of a move from the player A. To ensure that the set of winning strategies of the player A does not grow, the idea is to force every position outside of V_B to admit a winning strategy for the player B by *flooding* the language X with all words outside of V_B.

Proposition 9 *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, \Sigma^\star, V_B)$ be a cutenation game over the alphabet Σ with associated language $\mathcal{L}$. If V_B is closed under subword then the centralizer $\mathcal{C}(X \cup \Sigma^\star \setminus V_B)$ is equal to $\mathcal{L}$.*

PROOF. We prove that the player B has a winning strategy in the cutenation game if and only if the player B has a winning strategy in the commutation game of $X \cup \Sigma^\star \setminus V_B$.

If the player A had a winning strategy in the original game starting from a given position then he keeps playing according to his original strategy. As V_B

is closed under subword, starting from a word in V_B the player B cannot use a transition in $\Sigma^\star \setminus V_B$ to cut: the player B has exactly the same possible moves as in the original game. Therefore, the player A still has a winning strategy in the new game.

If the player B had a winning strategy in the original game starting from a given position then he keeps playing according to its original strategy as long as the player A keeps using moves that were valid in the original game. If the player A uses a new move then, as V_B is closed under subword, just after this move the new position is a word in $\Sigma^\star \setminus V_B$. A winning strategy for the player B starting from a word x in $\Sigma^\star \setminus V_B$ is simply to cut x completely thus accessing to the empty word position. The empty word position is winning for the player B: when the player A catenates a word y, the player B just cuts y, coming back to the empty word. Therefore, the player B still has a winning strategy in the new game. □

To combine both checking and flooding to remove the constraints on a game it is sufficient to ensure that the original constraints V_B are closed under subword.

Lemma 10 *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V_A, V_B)$ be a cutenation game over the alphabet Σ with associated language $\mathfrak{L}$ such that V_B is closed under subword. Let c be a new letter not in Σ, let $X' = X \cup cV_A^\star \cup V_A^\star c$ and $V'_B = V_B \cup cV_B \cup V_B c$. If the four sets $X^{-1}X^{-1}V_B$, $V_B X^{-1}X^{-1}$, $((X^{-1}(V_A X \cap V_B)) \setminus V_A) \cap V_A^\star$, and $(((XV_A \cap V_B)X^{-1}) \setminus V_A) \cap V_A^\star$ are empty then $\mathfrak{L}$ is equal to*

$$\mathcal{C}(X \cup cV_A^\star \cup V_A^\star c \cup (\Sigma \cup \{c\})^\star \setminus (V_B \cup cV_B \cup V_B c)) \cap \Sigma^\star \quad .$$

PROOF. If V_B is closed under subword, so is $V'_B = V_B \cup cV_B \cup V_B c$. To conclude, combine both Prop. 8 and Prop. 9. □

5 Gluing all together

We can now conclude the proof of the main statement by combining the three parts of the construction together.

Theorem 11 *There exists a rational language X whose centralizer $\mathcal{C}(X)$ is complete for co-recursively enumerable languages.*

PROOF. Let $\mathcal{P}$ be a Post tag sytem, whose language is complete for recursively enumerable languages. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ be the unfair rooted ori-

ented cutenation game obtained by Prop. 4 and such that for some $\mathfrak{a} \in \mathfrak{A}$ the equality $\mathfrak{L}(\mathfrak{a}) = \Sigma^\star \setminus L_{\mathcal{P}}$ holds. Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V'_A, V'_B)$ be the cutenation game with language $\mathfrak{L}$ obtained by Prop. 5 and Lemma 7 such that the equation $\mathfrak{L} \cap \{\langle s, x, t\rangle, x \in \Sigma^\star\} = \{\langle s, x, t\rangle, x \in \Sigma^\star \setminus L_{\mathcal{P}}\}$ holds for some $(s, t) \in \mathfrak{A}$. To combine this cutenation game with Lemma 10, as V'_B is closed under subword it is sufficient to show that the hypotheses of Prop. 8 are satisfied. More precisely, it is sufficient to show that the four sets $X^{-1}X^{-1}V'_B$, $V'_B X^{-1}X^{-1}$, $((X^{-1}(V'_A X \cap V'_B)) \setminus V'_A) \cap V'^\star_A$, and $(((XV'_A \cap V'_B)X^{-1}) \setminus V'_A) \cap V'^\star_A$ are empty and both inclusions $X^{-1}V'_B \subseteq V'_B$ and $V'_B X^{-1} \subseteq V'_B$ hold. Both inclusions hold because V'_B is closed under subword.

The set $X^{-1}X^{-1}V'_B$ is empty because first $\psi_R(R)^{-1}V'_B$ only contains the empty word and X does not contain the empty word, secondly because $\psi_L(L)^{-1}V'_B$ contains only the empty word and words which begin with a letter in $\Gamma_L^- \cup \{o\}$ and words in X never begin with such a letter. Symmetrically, the set $V'_B X^{-1}X^{-1}$ is empty.

The set $((X^{-1}(V'_A X \cap V'_B)) \setminus V'_A) \cap V'^\star_A$ is empty because all words in the language $X^{-1}(V'_A X \cap V'_B)$ contain exactly one occurence of a letter from $\Gamma_L^+ \cup \Gamma_R^+$. Symmetrically, the set $(((XV'_A \cap V'_B)X^{-1}) \setminus V'_A) \cap V'^\star_A$ is empty.

Therefore, Lemma 10 can be applied and $\Sigma^\star \setminus L_{\mathcal{P}}$ can be recursively computed from the centralizer of the rational set $X \cup cV'^\star_A \cup V'^\star_A c \cup (\Sigma' \cup \{c\})^\star \setminus (V'_B \cup cV'_B \cup V'_B c)$. As a consequence, the centralizer of this rational language is complete for co-recursively enumerable languages. □

References

[1] C. Choffrut, J. Karhumäki, N. Ollinger, The commutation of finite sets: a challenging problem, Theoret. Comput. Sci. 273 (2002) 69–79.

[2] J. H. Conway, Regular Algebra and Finite Machines, Chapman Hall, 1971.

[3] B. Ratoandramanana, Codes et motifs, RAIRO Theor. Informat. 23 (1989) 425–444.

[4] J. Karhumäki, M. Latteux, I. Petre, The commutation with codes and ternary sets of words, Theoret. Comput. Sci. 340 (2005) 322–333.

[5] J. Karhumäki, I. Petre, Conway's problem for three-word sets, Theoret. Comput. Sci. 289 (2002) 705–725.

[6] A. Okhotin, Decision problems for language equations with boolean operations, in: Proc. of ICALP 2003, Vol. 2719 of LNCS, Springer, 2003, pp. 239–251.

[7] J. Karhumäki, Challenges of commutation: an advertisement, in: Proc. of FCT 2001, Vol. 2138 of LNCS, Springer, 2001, pp. 15–23.

[8] T. Harju, O. Ibarra, J. Karhumäki, A. Salomaa, Decision questions in semilinearity and commutation, J. Comput. Syst. Sci. 65 (2002) 278–294.

[9] I. Petre, Commutation problems on sets of words and formal power series, Ph.D. thesis, University of Turku (2002).

[10] J. Karhumäki, I. Petre, Two problems on commutation of languages, in: G. Rozenberg, A. Salomaa (Eds.), Current Trends in Theoretical Computer Science, World Scientific, 2004.

[11] M. Kunc, Regular solutions of language inequalities and well quasi-orders, in: Proc. of ICALP 2004, Vol. 3142 of LNCS, Springer, 2004, pp. 870–881.

[12] M. Kunc, The power of commuting with finite sets of words, in: Proc. of STACS 2005, Vol. 3404 of LNCS, Springer, 2005, pp. 569–580.

[13] M. Kunc, The power of commuting with finite sets of words, Theory of Computing Systems 40 (4) (2007) 521–551.

[14] M. Minsky, Computation: Finite and Infinite Machines, Prentice Hall, 1967.